Floating Point Words using the Atari 8bit Math ROM#

General Information

Author: Carsten Strotmann
Language: FORTH
Compiler/Interpreter: volksForth
Published: Januar 2007

Usage#

The Floating Point words are in the vocabulary "FMATH". To use these word, put "FMATH" in the vocabulary searchlist: `FMATH ALSO`

Additional information about the Atari 8bit Math ROM can be found in De-Re Atari, Chapter 8 and Atari Reference Manual, Chapter 11.

All floating Point numbers are stored in 3 cells (2 byte, 6 byte in total).

Glossary

Constants

FR0 #

Address of 1st floating point pseudo register

FR1 #

Address of 2nd floating point pseudo register

FLPTR #

Address of 2 byte pointer to user buffer for Foating point number (not used by the definitions below)

INBUFF #

Address of 2 byte pointer to ASCII Foating point number

CIX #

Address of index into INBUFF

Floating Point Stack words#

F@ (addr -- fp)#

fetch floating point number (3 cells) stored at Address "addr" and place the on the data stack.
Example: `FR0 F@`

F! (fp addr --)#

store floating point number "fp" (3 cells) at Address "addr".

FSWAP (fp1 fp2 -- fp2 fp1)#

swap two floating point numbers on the stack (3 cells each)

FDROP (fp --)#

remove floating point number from top of stack (3 cells)

FDUP (fp -- fp fp)#

duplicate topmost floating point number on stack

FOVER (fp1 fp2 -- fp1 fp2 fp1)#

copy 2nd topmost floating point number to top of stack

F.TY (--)#

print floating point number (ASCII representation) already in Buffer referenced by INBUFF

F. (fp --)#

print floating point number on top of stack

F? (addr --)#

print floating point number at Address "addr"

Floating Point conversation#

FLOAT (n -- fp) #

convert integer number "n" to floating point numner "fp"

FIX (fp -- n)#

convert fix part of floating point number "fp" to integer number "n"

ASCF (addr -- fp)#

convert ASCII Floating Point number at "addr" (terminated by "zero" or Atari EOL 155/$9B) to floating point number "fp"

Floating Point Comparison#

F0= (fp -- f)#

Flag "f" is "true" if "fp" is equal zero "0"

F= (fp1 fp2 -- f)#

Flag "f" is "true" if "fp1" and "fp2" are equal

F< (fp1 fp2 -- f)#

Flag "f" is "true" if "fp2" is smaller than "fp1". The opposite comparison can be defined by `: F> FSWAP F< ;`

Floating Point Arithmetics#

F+ (fp1 fp2 -- fpn)#

add two floating point numbers on stack, leaving the result on stack

F- (fp1 fp2 -- fpn)#

substract two floating point numbers on stack, leaving the result on stack

F* (fp1 fp2 -- fpn)#

multiplicate two floating point numbers on stack, leaving the result on stack

F/ (fp1 fp2 -- fpn)#

divide fp1 by fp2, leaving the result on stack

FLOG (fp1 -- fplog)#

calculate natural logarithm of fp1

FLOG10 (fp1 -- fplog10)#

calculate base 10 logarithm of fp1

FEXP (fp1 -- fpexp)#

calculate natural exponentation of fp1

FEXP10 (fp1 -- fpexp10)#

calculate base 10 exponentation of fp1

Floating Point Forth Compiler Extension#

F, (fp --)#

store floating point number "fp" as 3 cells (6 bytes) into the directory

FCONSTANT (fp --)#

create a floating point constant. Example: `FP 1.234 FCONSTANT FCONST`

FVARIABLE (--)#

create a floating point variable. Example: `FVARIABLE FVAR FP 1.234 FVAR F!`

FP (-- fp)#

covert next word in input stream to floating point value on stack. Example: `FP 1.234 F.`

FLOATING (--)#

covert next word in input stream to floating point and compile as a literal into the current definition. Example: `: FLOATTEST FLOATING 1.234 F.`

FLITERAL (--)#

compiler word used to compile a floating point number into an defintion

```
\ Floating Point Extension
\ using Atari 8bit ROM FP Routines
\ based on FIG Forth APX20029

\needs CALL INCLUDE" D:CALL.FS"

CR .( loading Floating Point ext. )

VOCABULARY FMATH
FMATH ALSO DEFINITIONS

$D4 CONSTANT FR0
$E0 CONSTANT FR1
$FC CONSTANT FLPTR
$F3 CONSTANT INBUF
$F2 CONSTANT CIX

| : XCALL CALL DROP ;

| : AFP    $D800 XCALL ;
| : FASC   $D8E6 XCALL ;
| : IFP    $D9AA XCALL ;
| : FPI    $D9D2 XCALL ;
| : FADD   $DA66 XCALL ;
| : FSUB   $DA60 XCALL ;
| : FMUL   $DADB XCALL ;
| : FDIV   $DB28 XCALL ;
| : FLG    $DECD XCALL ;
| : FLG10  $DED1 XCALL ;
| : FEX    $DDC0 XCALL ;
| : FEX10  $DDCC XCALL ;
| : FPOLY  $DD40 XCALL ;

: F@ ( addr -- fp )
  >R R@ @ R@ 2+ @ R> 4 + @ ;

: F! ( fp addr -- )
  >R R@ 4 + ! R@ 2+ ! R> ! ;

: F.TY ( -- )
  BEGIN
    INBUF @ C@ DUP $7F AND EMIT
    1 INBUF +!
  $80 > UNTIL ;

: FSWAP ( fp1 fp2 -- fp2 fp1 )
```

```
  5 ROLL 5 ROLL 5 ROLL ;

: FDROP ( fp -- )
  2DROP DROP ;

: FDUP  ( fp -- fp fp )
  2 PICK 2 PICK 2 PICK ;

: FOVER ( fp1 fp2 -- fp1 fp2 fp1 )
  5 PICK 5 PICK 5 PICK ;

: F. ( fp -- )
  FR0 F@ FSWAP FR0 F!
  FASC F.TY SPACE
  FR0 F! ;

: F? ( addr -- )
  F@ F. ;

: <F ( fp1 fp2 -- )
  FR1 F! FR0 F! ;

: F> ( -- fp1 )
  FR0 F@ ;

: FS ( fp -- )
  FR0 F! ;

: F+ <F FADD F> ;
: F- <F FSUB F> ;
: F* <F FMUL F> ;
: F/ <F FDIV F> ;

: FLOAT ( n -- fp )
  FR0 ! IFP F> ;

: FIX   ( fp -- n )
  FS FPI FR0 @ ;

: FLOG    FS FLG   F> ;
: FLOG10  FS FLG10 F> ;
: FEXP    FS FEX   F> ;
: FEXP10  FS FEX10 F> ;

: ASCF ( addr -- fp )
  INBUF ! 0 CIX C! AFP F> ;

: F0= OR OR 0= ;
: F=  F- F0= ;
: F<  F- 2DROP $80 AND 0 > ;

: F, ( fp -- )
  ROT , SWAP , , ;

: FCONSTANT
  CREATE F, DOES> F@ ;

: FVARIABLE
  CREATE 6 ALLOT DOES> ;
```

```
| : FLIT
    R> DUP 6 + >R F@ ;

: FLITERAL
  COMPILE FLIT F, ;

: FP ( -- fp )
  BL WORD 1+ ASCF ; IMMEDIATE

: FLOATING
  BL WORD 1+
  ASCF FLITERAL ; IMMEDIATE

: [FLOATING] [COMPILE] FLOATING ; IMMEDIATE

CR .( Floating Point ext. loaded. ) CR

ONLYFORTH
```